อาจหาญ สัตยารักษ์

ภาษาไทยกราฟิก
รหัส สมอ.

ศึกษาเทคนิคการเขียนโปรแกรมฟอนต์กราฟิกภาษาไทยด้วยรหัส สมอ.

ในบทความที่ผ่านมาของผู้เขียนได้เขียนภาษาไทยกราฟิกไว้มากมาย แต่ส่วนใหญ่จะเป็นรหัสเกษตร ยังไม่เคยเสนอแนวคิดการสร้างภาษาไทยรหัส สมอ.มาก่อนเลย หลังจากที่ผู้เขียนได้ออกหนังสือเทคนิคการเขียนกราฟิกบนเทอร์โบปาสคาลไม่นาน ก็มีนิสิตที่ต้องพัฒนาโปรแกรมโดยใช้ภาษาไทยรหัส สมอ. เขียนไปถามในส่วนการทำภาษาไทยกราฟิกของรหัสดังกล่าว

ผู้เขียนคิดว่าคงเป็นประโยชน์ต่อผู้อ่านท่านอื่น ๆ ด้วย จึงได้เขียนบทความนี้ขึ้นมา โดยยังคงจัดทำเป็นซอฟต์เมล์เช่นเดิมในชื่อไฟล์ว่า thaitis.pas ซึ่งเป็นยูนิตไฟล์ของระบบภาษาไทยแบบนี้กับ testtis.pas เพื่อเป็นไฟล์ตัวอย่างการแสดงข้อความบนจอภาพด้วยยูนิตนี้

ฟอนต์อักขระภาษาไทย

ปัญหาเรื่องฟอนต์ภาษาไทยนี้ผู้เขียนได้เคยเขียนถึงหลายครั้งแล้ว ทั้งในวารสารไมโครคอมพิวเตอร์และหนังสือที่ผู้เขียนแต่ง เช่น เทคนิคการเขียนกราฟิกบนเทอร์โบปาสคาล หน้า 59 (หนังสือพิมพ์มีจำหน่ายที่บริษัทซีเอ็ดยูเคชั่น จำกัด) ก็ได้เขียนไว้เช่นกัน

ซึ่งมีปัญหาที่ผู้อ่านหลายท่านถามไปยังผู้เขียน จึงขอนำมากล่าวถึงพอสังเขปดังต่อไปนี้

1. จุดสำคัญของการสร้างภาษาไทยกราฟิกไม่ได้อยู่ตรงไฟล์ THAIRW.COM หรือการใช้ PCTOOLS ตรวจดูไฟล์แต่อยู่ตรงสองบรรทัดสุดท้ายของหน้านั้น คือ โดยแต่ละอักขระจะใช้ 4080/255=16 ไบต์ ซึ่งเป็นการแสดงให้เห็นว่า อักษรเหล่านี้เกิดจากการสร้างรูปแบบให้ตรงกับภาพของอักขระแต่ละตัวในขนาด 16 ไบต์

2. ชุดคำสั่ง ShowChar ในหน้า 60 จะสัมพันธ์กัน คือ เขียนในช่วง k=0 ถึง 15 หรือ 16 ไบต์ และตัวเลขนำมาตรวจตำแหน่งของแต่ละอักขระต้องคูณด้วย 16 คือ (ASCII-1)*16 ซึ่งมีอยู่ 3 ที่

ด้วยกัน สองจุดนี้แหละครับเป็นจุดใหญ่ของการสร้างภาษาไทย

เมื่อเราทราบจุดสำคัญสองจุดนี้แล้ว เราก็สามารถปรับปรุงโปรแกรมให้ใช้กับไฟล์ข้อมูลอื่นได้ เช่น ราชวิถีเวิร์ดพีซีมีไฟล์ชื่อ RVGA.COM ซึ่งแสดงอักขระภาษาไทยเช่นเดียวกัน แต่มีเฉพาะตัวอักขระภาษาไทยเท่านั้น เพราะภาษาอังกฤษใช้ของเดิม หรือไฟล์ font1.csl, font1.csl ก็สามารถจะนำมาใช้งานได้เช่นกัน โดยไฟล์ *.csl นี้จะมีขนาดไฟล์ 6144 ไบต์ จึงใช้อักขระละ 6144/256=24 ไบต์

หากเรากำหนดตัวเลขให้ตรงกับขนาดของไฟล์คือ เปลี่ยน 16 เป็น 24 และเปลี่ยน 15 เป็น 23 เปลี่ยน 4080 เป็น 6144 ก็สามารถนำมาสร้างภาษาไทยกราฟิกได้เช่นกัน แต่เนื่องจากไฟล์ *.csl นี้จะมีลำดับข้อมูลตรงกับรหัสแอสกี ดังนั้นจึงต้องเปลี่ยน ASCII-1 ให้เป็น ASCII เท่านั้น โดยจะต้องปรับแต่งการแสดงผลของสระและวรรณยุกต์อีกเล็กน้อย

แต่เนื่องจากไฟล์ *.csl นี้เก็บลำดับข้อมูลตามรหัสเกษตร จึงไม่อยู่ในส่วนที่บทความนี้จะเขียนถึง ลองหันมาดูไฟล์อักขระใน CU Writer กันบ้าง ใน CU Writer มีไฟล์ข้อมูลอักขระที่ใช้แสดงบนจอภาพอยู่ 2 ไฟล์ คือ ไฟล์ normal.fon (ได้ตัวปกติ) และ italic.fon (ได้ตัวเอียง) ก็นำมาใช้แสดงตัวอักขระกราฟิกบนจอภาพตามที่บทความต้องการได้ ถ้าดูรายละเอียดของไฟล์ทั้งสองจะเป็นดังนี้

1. ความยาวของไฟล์เป็น 5120 ไบต์

2. ลองเอา 256 หารจะได้ 20 ไบต์ แสดงว่าหนึ่งอักขระใช้ 20 ไบต์

หมายเหตุ

รหัสแอสกีจะมีทั้งหมด 256 ตัว แต่ไฟล์ที่อ้างถึงในหนังสือนั้นเขียนเพียง 255 ตัวคือจาก 0 ถึง 254 เวลาอ้างถึงจึงต้องเป็น (ASCII-1)*16 แต่ของ CU นั้นเขียนจาก 0 ถึง 255 จึงมี 256 ตัว เวลาอ้างถึงจึงเป็นแค่ ASCII*20 เท่านั้น

แต่เนื่องจากไฟล์อักขระทั้งสองไฟล์นี้เรียงลำดับตามรหัส สมอ. ดังนั้นชุดคำสั่งแสดงข้อความบนจอภาพจึงต้องเขียนขึ้นใหม่ด้วย จึงจะได้ผลบนจอภาพที่ถูกต้อง

วันนี้เราจะมีศึกษาการเขียนภาษาไทยกราฟิกด้วยรหัส สมอ.กัน ผู้เขียนจึงเรียนขออนุญาตจากคณะวิศวกรรมศาสตร์ จุฬาฯ เจ้าของเวิร์คจุฬามาพร้อมนี้ด้วย

ตารางรหัสแอสกีภาษาไทย

ก่อนที่เราจะเขียนภาษาไทยกราฟิกนั้น สิ่งที่ต้องรู้คือรหัสแอสกีใดเป็นของอักขระใด ทั้งนี้เพราะว่าการแสดงภาษาไทยมี 3 ระดับ อีกทั้งการเขียนภาษาไทยไม่ใช่วางเรียงลำดับแบบต่อกันไป แต่จะมีการขยับไปเขียนบนตัวอักขระหรือล่างตัวอักขระที่เขียนไปก่อนแล้ว

ขั้นตอนแรกจึงต้องมาดูภาษาไทยตามรหัส สมอ. ก่อน ซึ่งจะได้ดังตารางในรูปที่ 1

	0	1	2	3	4	5	6	7	8	9
16		ก	ข	ฃ	ค	ฅ	ฆ	ง	จ	ฉ
17	ช	ซ	ฌ	ญ	ฎ	ฏ	ฐ	ฑ	ฒ	ณ
18	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ
19	พ	ฟ	ภ	ม	ย	ร	ฤ	ล	ภ	ว
20	ศ	ษ	ส	ห	ฬ	อ	ฮ	ฯ	ะ	ั
21	า	ำ	ิ	ี	ึ	ื	ุ	ู		ฺ
22					เ	แ	โ	ใ	ไ	
23	ๅ	็	่	้	๊	๋	์	๎		
24	๐	๑	๒	๓	๔	๕	๖	๗	๘	๙

ตารางแสดง
ภาษาไทยรหัส สมอ.
จากไฟล์ NORMAL.FON
ของ CU Writer

รูปที่ 1 แสดงตารางส่วนของภาษาไทยที่อ่านได้จากไฟล์ NORMAL.FON

พิจารณารหัสแอสกีตามตาราง (เฉพาะภาษาไทย) จะพบว่า

1. อักขระที่อยู่ระดับล่าง ได้แก่ รหัสแอสกีที่ 216,217,218

2. อักขระที่อยู่ระดับบน ได้แก่ รหัสแอสกีที่ 209,212-215,219,231-238

3. อักขระที่อยู่ระดับกลาง คือ อักขระที่ไม่ได้กล่าวถึงในข้อ 1 และ 2 ทั้งหมด

เนื่องจากไฟล์อักขระกำหนดความสูงไว้ถึง 20 จุด (จากข้อมูลของอักขระละ 20 ไบต์) ดังนั้นเราจึงไม่ต้องกำหนดการเลื่อนขึ้นหรือลง เพียงแต่สั่งให้แสดงอักขระนั้น ๆ ถอยหลังก็จะไม่เขียนทับอักขระเดิมแล้ว

เพื่อให้การแสดงอักขระถูกต้องเราก็จะตรวจสอบว่ากำลังวาดภาพอักขระใด ถ้าเป็นการวาดอักขระในข้อ 1 และ 2 ให้เลื่อนไปแสดงที่ตำแหน่งถอยหลังโดยยังคงจุดขวาสุดไว้ที่ตำแหน่งเดิม เพื่อที่จะได้เขียนอักขระตัวต่อไปได้ในตำแหน่งถัดไปทางขวา

การแสดงอักขระบนจอภาพซึ่งเป็นอักขระที่ไม่โตเกินไปนี้ มีความยุ่งยากน้อยกว่าการกำหนดในการพิมพ์ลงกระดาษ ทั้งนี้เพราะภาษาไทยมีตัวอักขระที่ไม่ได้อยู่ในกรอบสี่เหลี่ยมเสมอไปประการหนึ่ง กับการแสดงผล 3 ระดับอีกประการหนึ่ง ถ้าท่านสังเกตตัวอักขระที่พิมพ์จากเครื่องพิมพ์เลเซอร์ให้ดี อาจพบความผิดปกติได้ โดยเฉพาะเมื่อพิมพ์ตัวขนาดใหญ่ คำที่ผู้เขียนแนะให้สังเกตก็คือ นึก ณี ฉิ่ง ยี จะพบว่าตำแหน่งของสระอยู่ผิดที่อย่างน่าเกลียดมาก ๆ การที่เป็นอย่างนี้เพราะว่าไฟล์เหล่านี้ส่วนใหญ่จะตรวจตำแหน่งขวาสุดเพียงตำแหน่งเดียว ซึ่งในความเห็นของผู้เขียนคิดว่า ในการพิมพ์ตัวขนาดใหญ่จากเครื่องเลเซอร์นี้น่าจะมีจุดขวาสุด 2 จุดโดยคำนึงถึงตัวอักขระบางตัวเช่น ณ, น, ฉ, ษ ฯลฯ ซึ่งต้องมีขวาสุดดังนี้

จุดขวาสุดจุดแรก เพื่อเป็นตำแหน่งที่จะนำอักขระระดับกลางมาแสดงต่อไปทางขวามือ

จุดขวาสุดจุดที่สอง เพื่อเป็นตำแหน่งที่ให้สระหรือวรรณยุกต์มาแสดงที่ข้างบนหรือข้างล่าง

ทั้งนี้เพราะตัวอักขระพวกนี้ ขวาสุดของการแสดงสองแบบข้างต้นนี้ไม่ใช่จุดเดียวกัน จึงขอฝากความคิดเห็นนี้ไปยังผู้พัฒนาโปรแกรมพิมพ์ในเครื่องเลเซอร์นำไปแก้ไขด้วย (ผู้เขียนเองมีเพียงแนวคิดเท่านั้น แต่คงไม่มีทางได้เขียนโปรแกรมบนเครื่องเลเซอร์เพราะขาดปัจจัยสำคัญที่จะซื้อมาใช้งาน)

สร้างยูนิตภาษาไทย รหัส สมอ.

เพื่อสะดวกในการเรียกใช้ในโปรแกรมต่อไป เราจึงจะสร้างระบบภาษาไทยนี้เป็นยูนิตไฟล์ โดยมีการกำหนดชุดคำสั่งที่จำเป็นเรียกใช้ได้ไว้ล่วงหน้าด้วย การสร้างระบบภาษาไทยนี้จะยึดหลักดังนี้

1. ให้ยูนิตเข้าสู่กราฟิกโหมดโดยอัตโนมัติ

2. อ่านข้อมูลของอักขระที่ต้องการ

```
 1: {================ THAITIS.PAS ====================}
 2: UNIT ThaiTIS;
 3: interface
 4: uses dos,crt,graph;
 5: Var Mono,VgaMono:boolean;
 6:     ch,Funckey:char;
 7: PROCEDURE outthaixy(column,row:integer; content:string);
 8: FUNCTION  SwapBit(b:byte):byte;
 9: PROCEDURE ReadChar;
10: PROCEDURE Set_Color(Color:word);
11: implementation

12: var  font:array[0..1,0..5119] of byte;
13:      NoFont:byte;
14:      Enlarge:byte;

15: PROCEDURE Opengraph;
16: var grdriver;grmode:integer;
17: begin
18:    Mono:=false; VgaMono:=false;
19:    detectgraph(grdriver,grmode);
20:    case grdriver of
21:      5:VgaMono:=true;
22:      7:Mono:=true;
23:      9:grmode:=1;
24:    end;
25:    initgraph(grdriver,grmode,'');
26: end;
```

รูปที่ 2 ยูนิต Thaitis.tpu แสดงส่วนหัวและชุดคำสั่งเข้าสู่กราฟิกโหมด

มาไว้ในหน่วยความจำทันทีที่เรียกใช้ยูนิต

3. ต้องทำงานได้ทั้งจอโมโนโครม และจอวีจีเอหรือวีจีเอโมโนโครม

4. ข้อความที่ปรากฏบนจอภาพต้องกำหนดสีได้ด้วยการควบคุมด้วย Ctrl code

5. ข้อความจะต้องสามารถสร้างแบบตัวขยายหรือแบบปกติได้โดยง่าย

นำแนวคิดนี้มาเขียนไฟล์ชื่อ thaitis.pas เพื่อคอมพายล์เป็นไฟล์ *.tpu ได้ โดยการกำหนดส่วนสำคัญดังรูปที่ 2 มีรายละเอียดดังนี้

1. บรรทัดที่ 2 กำหนดว่าเป็นไฟล์ยูนิต (ต้องเป็นชื่อเดียวกันกับไฟล์ด้วย)

2. บรรทัดที่ 4-6 กำหนดยูนิตที่นำมาใช้งานร่วมและกำหนดตัวแปรใช้งาน

3. บรรทัดที่ 7-10 ชุดคำสั่งหรือฟังก์ชันที่สามารถเรียกใช้ได้จากภายนอกยูนิต

4. บรรทัดที่ 12 กำหนดตัวแปรรองรับข้อมูลอักขระ 2 ฟอนต์คือตัวปกติและตัวเอียง

5. บรรทัดที่ 13-14 ตัวแปรเพื่อกำหนดว่าใช้ฟอนต์ใด NoFont=0 ตัวปกติ NoFont=1 ตัวเอียง และกำหนดว่าเป็นตัวขยายหรือตัวปกติ Enlarge=1 ตัวขยาย

6. บรรทัดที่ 15-26 ชุดคำสั่งเข้ากราฟิกแบบอาศัยไฟล์ *.bgi (หากต้องการเข้ากราฟิกแบบไม่ใช้ไฟล์ *.bgi ให้หาอ่านได้จากหนังสือ เทคนิคการเขียนกราฟิกบนเทอร์โบปาสคาล)

```
27: FUNCTION SwapBit(b:byte):byte;
28: begin
29:    swapbit:=b and 1 shl 7  + b and 2 shl 5   +
30:             b and 4 shl 3  + b and 8 shl 1   +
31:             b and 16 shr 1 + b and 32 shr 3  +
32:             b and 64 shr 5 + b and 128 shr 7:
33: end;

34: FUNCTION LowBitToByte(B:Byte):byte;
35: begin
36:  LowBitToByte:=Lo(B) and 8 shl 4 + Lo(B) and 8 shl 3 +
37:                Lo(B) and 4 shl 3 + Lo(B) and 4 shl 2 +
38:                Lo(B) and 2 shl 2 + Lo(B) and 2 shl 1 +
39:                Lo(B) and 1 shl 1 + Lo(B) and 1;
40: end;

41: FUNCTION ByteToWord(B:Byte):word;
42: begin
43:   ByteTOWord:=(LowBitToByte(B shr 4))*256+LowBitToByte(B);
44: end;

45: PROCEDURE ReadChar;
46: var regs:registers;
47: begin
48:   regs.AH:=$00; intr($16,regs);
49:   ch:=chr(regs.AL);
50:   if ch=#0 then Funckey:=Chr(regs.AH);
51: end;

52: PROCEDURE Set_Color(Color:word);
53: begin
54:    setcolor(color);
55:    if mono then setcolor(1);
56:    if Vgamono then setcolor(3);
57: end;
```

รูปที่ 3 ฟังก์ชันในการเปลี่ยนค่าประจำบิตที่ใช้งาน ชุดคำสั่งอ่านการกดแป้นและกำหนดสี

การสลับค่าบิตและการขยายบิต

เนื่องจากในการแสดงอักขระบนจอภาพที่จะนำเสนอต่อไปนี้เป็นการใช้คำสั่งในการลากเส้นตรง คือ คำสั่ง line โดยใช้การกำหนดรูปแบบของเส้นตรงตามรูปแบบของตัวอักขระที่ต้องการก่อนคำสั่งที่ช่วยกำหนดเส้นตรงให้เป็นรูปแบบที่ต้องการคือคำสั่ง SetLineStyle

ไฟล์อักขระที่จะใช้คือ normal.fon และ italic.fon จากโปรแกรมเวิร์ดจุฬานี้ กำหนดรูปแบบสำหรับจอโมโนโครม ซึ่งจะให้รูปแบบเส้นที่แตกต่างจากจอสีหรือจอวีจีเอโมโนโครม (คือถ้าหากจัดการไม่ถูกต้องจะได้ภาพอักขระคล้ายกับมองผ่านกระจกเงา) ดังนั้นส่วนสำคัญอย่างหนึ่งที่เราต้องทำคือ หากใช้ไฟล์นี้กับจอสีหรือวีจีเอจะต้องคำนวณข้อมูลใหม่

การแสดงอักขระในยุคปัจจุบันมักจะแสดงตัวปกติและตัวขยายได้ ดังนั้นการคำนวณเพื่อให้ได้อักขระแบบขยายจึงทำให้เกิดชุดคำสั่งในเรื่องนี้ดังรูปที่ 3 มีรายละเอียดดังนี้

1. บรรทัดที่ 27-33 ฟังก์ชันสลับค่าประจำบิต เพื่อให้อักขระที่กลับซ้ายขวาเมื่อแสดงบนจอวีจีเอแสดงได้ถูกต้อง

หลักการคือ สลับข้อมูลระหว่างบิตที่ 7 กับบิตที่ 0, บิตที่ 6 กับบิตที่ 1, บิตที่ 5 กับบิตที่ 2 และบิตที่ 4 กับบิตที่ 3 โดยการตรวจลงบิตด้วย and และเลื่อนค่าด้วย shl หรือ shr

2. บรรทัดที่ 34-40 ฟังก์ชันการขยายสี่บิตต่ำของหนึ่งไบต์ไปเป็นข้อมูลหนึ่งไบต์ คือขยายบิตที่ 3, 2, 1, 0 ให้ไปเป็นบิตที่ 7-6, 5-4, 3-2, 1-0 ตามลำดับ ซึ่งจะทำให้ได้ข้อมูลหนึ่งไบต์มาใช้งาน

3. บรรทัดที่ 41-44 ฟังก์ชันขยายข้อมูลหนึ่งไบต์ไปเป็นข้อมูลสองไบต์ (คือได้ข้อมูลชนิด word มา) ใช้หลักการนำเอาสี่บิตสูงมาเลื่อนค่าให้เป็นสี่บิตต่ำแล้วขยายให้เป็นข้อมูลหนึ่งไบต์นำข้อมูลที่ได้นี้คูณด้วย 256 (หรือจะใช้คำสั่ง shl 8 ก็ได้) จากนั้นนำมาบวกกับข้อมูลสี่บิตต่ำที่ขยายให้เป็นข้อมูลหนึ่งไบต์ ซึ่งจากฟังก์ชันนี้จะทำให้คำสั่งกำหนดเส้นตรงจากข้อมูลหนึ่งไบต์ (ได้จุด 8 จุด) มาเป็นข้อมูล 2 ไบต์ (ได้จุด 16 จุด) ซึ่งก็คือการขยายอักขระจาก 8 จุด เป็น 16 จุด นำไปแสดงตัวอักขระแบบขยายได้

4. บรรทัดที่ 46-51 ชุดคำสั่งตรวจการกดแป้น จะได้ ch เก็บรหัสแอสกีของแป้นปกติและตัวแปร Funckey เก็บรหัสแป้นพิเศษ

5. บรรทัดที่ 52-57 ชุดคำสั่งลงสีเพื่อให้สามารถกำหนดสีได้ตรงกับชนิดของจอภาพ

อ่านข้อมูลมาใช้งาน

เมื่อเราได้ชุดคำสั่งและฟังก์ชันที่ช่วยงานได้ครบถ้วนแล้ว ก็มาถึงขั้นตอนการอ่านข้อมูลของอักขระในไฟล์ *.fon เข้ามาสู่หน่วยความจำ โดยที่เราจะต้องเรียกใช้งานโดยการกำหนดรหัสที่ไม่ยุ่งยากมากนัก ทั้งต้องให้รูปอักขระบนจอภาพได้อย่างถูกต้องด้วย

```
58: PROCEDURE readfont;
59: var f:file; k:word;
60: begin
61:   assign(f,'Normal.fon');
62:   reset(f,1);
63:   blockread(f,font[0],5120);
64:   close (f);
65:   assign(f,'Italic.fon');
66:   reset(f,1);
67:   blockread(f,font[1],5120);
68:   close (f);
69:   if pred(Mono) then
70:    begin
71:     for k:=0 to 5119 do
72:       begin
73:         font[0][k]:=swapbit(Font[0][k]);
74:         font[1][k]:=swapbit(Font[1][k]);
75:       end;
76:    end;
77:    NoFont:=0; Enlarge:=0;
78: end;
```

รูปที่ 4
ชุดคำสั่งในการอ่านข้อมูลจากไฟล์ข้อมูลตัวอักขระ

เราจึงเขียนชุดคำสั่งในการอ่านไฟล์ได้ดังรูปที่ 4 มีรายละเอียดดังนี้

1. ตัวแปรที่รองรับข้อมูลคือ font[0] เก็บข้อมูลตัวปกติ และ font[1] เก็บข้อมูลตัวเอียง

2. บรรทัดที่ 61-64 เปิดไฟล์ normal.fon และอ่านข้อมูลสำหรับตัวปกติ

3. บรรทัดที่ 65-68 เปิดไฟล์ italic.fon และอ่านข้อมูลสำหรับตัวเอียง

4. บรรทัดที่ 69-76 ในกรณีที่ตรวจพบว่าไม่ใช่จอโมโนโครม (ตรวจจากชุดคำสั่ง opengraph;) ก็จะต้องนำข้อมูลที่อ่านได้ทั้งหมดมาทำการสลับค่าประจำบิตเสียก่อน

5. บรรทัดที่ 77 กำหนดให้เริ่มที่ตัวปกติ (Nofont=0) และเป็นตัวปกติ (Enlarge=0)

ชุดคำสั่งนี้จะถูกกำหนดไว้ในส่วนของโปรแกรมหลักจึงจะทำงานทันทีที่รันโปรแกรม

แสดงอักขระกราฟิกบนจอภาพ

เรื่องสุดท้ายในการสร้างยูนิตภาษาไทยรหัส สมอ. นี้ ก็คงเป็นการนำข้อมูลอักขระที่อ่านมาได้ไปทำให้เกิดภาพบนจอภาพในตำแหน่งที่กำหนดให้ โดยจะกำหนดการทำงานดังนี้

1. เริ่มต้นด้วยการแสดงตัวอักขระตามปกติด้วยฟอนต์ปกติก่อน

2. ใช้รหัสแอสกี 1 หรือ Ctrl-A หรือ #1 สำหรับกำหนดฟอนต์ปกติ

3. ใช้รหัสแอสกี 2 หรือ Ctrl-B หรือ #2 สำหรับกำหนดฟอนต์ตัวเอียง

4. ใช้รหัสแอสกี 3 หรือ Ctrl-C หรือ #3 สำหรับกำหนดสีของตัวอักษร โดยให้ป้อนตัวเลข 0-9 ตามหลัง Ctrl-C นี้ หากต้องการได้สีหมายเลข 10-15

รูปที่ 5
ชุดคำสั่งแสดงอักขระและข้อความ (รหัส สมอ.) บนจอภาพกราฟิก

```
79: PROCEDURE showchar(x,y:integer; ASCII:byte);
80: var k,FontShow:word;
81: begin
82:  for k:=0 to 19 do
83:   begin
84:    if font[NoFont][ASCII*20+k] <> 0  then
85:     begin
86:      if Enlarge=1 then FontShow:=ByteToWord(Font[NoFont][ASCII*20+k])
87:                   else FontShow:=font[NoFont][ASCII*20+k];
88:      setlinestyle(4,Fontshow,1);
89:      line(x,y+k,x+15,y+k);
90:     end;
91:   end;
92: end;

93: PROCEDURE outthaixy(column,row:integer; content:string);
94: var k,xpix,Ypix,c:word;
95: begin
96:  column:=(column-1)*8;  row:=(row-1)*16;
97:  Content:=Content+' ';
98:  for k:=1 to length(content) do
99:   begin
100:  repeat
101:     if content[k] in [#1,#2] then
102:       begin
103:          NoFont:=ord(content[k])-1;
104:          inc(k);
105:       end;
106:     if content[k]=#5 then
107:       begin
108:         Inc(Enlarge); inc(k);
109:         if Enlarge=2 then Enlarge:=0;
110:       end;
111:     if content[k]=#3 then
112:       begin
113:        inc(k);  c:=ord(content[k])-48;
114:        inc(k);
115:        if content[k]=#3 then
116:          begin
117:            inc(C,9); Inc(k);
118:          end;
119:        set_color(c);
120:       end;
121:  until content[k]>#31;
122:   xpix:=0;  Ypix:=0;
123:   if content[k] in [#209,#212..#219,#231..#238] then xpix:=8+Enlarge*8;
124:   if (content[k] in [#232..#237]) and
125:                 (content[k-1] in [#32..#206]) then ypix:=3;
126:   if (content[k] in [#209,#212..#215,#218,#219]) and
127:                      (NoFont=1) then dec(xpix,(Enlarge+1)*3);
128:   if (content[k] in [#231..#238]) and
129:                      (NoFont=1) then dec(xpix,(Enlarge+1)*4);
130:   if (content[k] in [#216,#217]) and
131:                      (NoFont=1) then inc(xpix,(Enlarge+1)*1);
132:   showchar(column-xpix,row+ypix,ord(content[k]));
133:   if content[k] in [#1..#208,#210..#211,#220..#230,#239..#255] then
```

รูปที่ 5 (ต่อ)

```
134:                                                    inc(column,8+Enlarge*8);
135:    end;
136:       setlinestyle(0,255,1);
137: end;

138: {********************** THAITIS.TPU ***************************}
139: begin
140:  opengraph;
141:  readfont;
142: end.
```

ให้ป้อน Ctrl-C ปิดท้ายอีกครั้งหนึ่ง

5. ใช้รหัสแอสกี 5 หรือ Ctrl-E หรือ #5 เพื่อเข้าออกตัวขยาย

หากท่านต้องการกำหนดตัวยกกำลัง ตัวห้อยท้าย หรือ ตัวขีดเส้นใต้ ก็ให้ลองพัฒนาเพิ่มเติมเองอีกที สามารถนำมาเขียนโปรแกรมได้ดังรูปที่ 5 มีรายละเอียดดังนี้

1. ชุดคำสั่ง showchar สำหรับแสดงอักขระครั้งละหนึ่งตัวบนจอภาพ มีรายละเอียดดังนี้

1.1 บรรทัดที่ 82 กำหนดการลากเส้น 20 เส้น จาก k=0 ถึง k=19

1.2 บรรทัดที่ 84-89 ในกรณีที่ข้อมูลลำดับนั้นมีค่าไม่เท่ากับศูนย์ก็จะมีการลากเส้นตรง

1.3 บรรทัดที่ 86-87 กำหนดตัวแปร Fontshow ให้ตรงกับขนาดอักขระที่กำหนด ถ้าเป็นตัวขยายก็จะให้ข้อมูลขนาดสองไบต์ แต่ถ้าเป็นตัวปกติก็จะใช้ข้อมูลหนึ่งไบต์ตามค่าที่อ่านได้จากไฟล์ข้อมูล

1.4 บรรทัดที่ 88 กำหนดรูปแบบของเส้นตรงตามรูปร่างของอักขระที่อ่านมาได้และลากเส้นขนาด 16 จุด แต่จะเกิดจุดตามข้อมูลที่กำหนด คือถ้า fontshow มีค่าหนึ่งไบต์ก็จะมีจุดเพียง 8 จุด แต่ถ้าได้ข้อมูลเป็นสองไบต์ก็จะมีจุด 16 จุด

2. ชุดคำสั่ง outThaixy เพื่อนำข้อความที่กำหนดไปแสดง ณ ตำแหน่งที่กำหนด มีรายละเอียดดังนี้

2.1 บรรทัดที่ 96 คำนวณหาตำแหน่งที่แสดงข้อความจริง (ค่าที่กำหนดให้มายึดแนวกับการแสดงแบบคอลัมน์และแถว) โดยกำหนดให้แต่ละตัวอักขระห่างกัน 8 จุด

2.2 บรรทัดที่ 101-105 ตรวจการกำหนดฟอนต์ว่าเป็นตัวปกติหรือตัวเอียง โดยใช้รหัสแอสกี 1 หรือ 2 ดังกล่าวมาแล้ว

2.3 บรรทัดที่ 106-110 ส่วนการตรวจว่าจะแสดงตัวขยายหรือตัวปกติ

2.4 บรรทัดที่ 111-120 ส่วนการตรวจสีของตัวอักขระด้วยการใช้ Ctrl-C และตัวเลข 0-9 ประกอบกัน เมื่อได้ค่าสีแล้วกำหนดสีด้วยชุดคำสั่ง set_color

2.5 บรรทัดที่ 122 กำหนดค่าการเลื่อนจุดให้เป็น 0

2.6 บรรทัดที่ 123 ในกรณีที่แสดงสระหรือวรรณยุกต์ ให้กำหนดค่าที่เลื่อนตำแหน่งไปตามค่าที่คำนวณได้ คือ xpix:=8+Enlarge*8;

2.7 บรรทัดที่ 124-125 กำหนดการเลื่อนค่าของวรรณยุกต์ให้ลงมา

3 จุดในกรณีที่ไม่ได้เขียนบนสระ เพื่อให้ได้ข้อความบนจอภาพที่สวยงาม

2.8 บรรทัดที่ 126-131 ในกรณีที่ใช้ตัวเอียงจะกำหนดตำแหน่งการแสดงสระ วรรณยุกต์ ให้มีตำแหน่งที่สวยงามขึ้น

2.9 บรรทัดที่ 132 เรียกใช้คำสั่งแสดงอักขระ

2.10 บรรทัดที่ 133-134 ถ้าเป็นอักขระระดับกลาง ก็ให้เลื่อนจุดแสดงอักขระตัวต่อไปไปทางขวาเท่าที่คำนวณได้ คือ inc(Column,8+Enlarge*8);

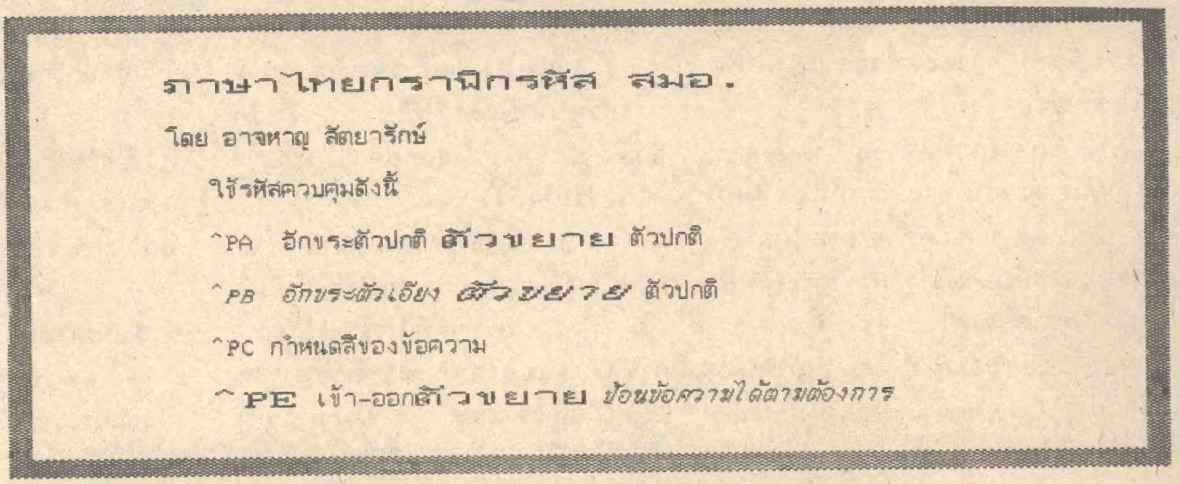
2.11 บรรทัดที่ 136 กำหนดรูปแบบเส้นให้เป็นเส้นทึบ

3. บรรทัดที่ 139-142 ส่วนของโปรแกรมหลัก ซึ่งกำหนดให้ยูนิตนี้ทำการเข้าสู่กราฟิกโหมด และทำการอ่านไฟล์ข้อมูลอักขระเข้ามาสู่หน่วยความจำ ซึ่งทำให้สะดวกในการเรียกใช้งานในไฟล์อื่น ๆ

รูปที่ 6
ไฟล์ testtis.pas แสดงตัวอย่างการนำข้อความ (รหัส สมอ.) ไปแสดงบนจอภาพ

```
1: {============= TESTTIS.PAS ================}
2: uses crt,graph,thaitis;
3: begin
4:  OutThaixy(12,3,#3'1'#3#5'ภาษาไทยกราฟิกรหัส สมอ.'#5);
5:  OutThaixy(12,5,#3'5'#3'โดย อาจหาญ สัตยารักษ์');
6:  OutThaixy(15,7,#3'3ใช้รหัสควบคุมดังนี้');
7:  OutThaixy(15,9,#1#3'6'#3'^PA  อักขระตัวปกติ '#5'ตัวขยาย'#5' ตัวปกติ');
8:  OutThaixy(15,11,#2'^PB  อักขระตัวเอียง '#5'ตัวขยาย'#5#1' ตัวปกติ');
9:  OutThaixy(15,13,'^PC กำหนดสีของข้อความ');
10:  OutThaixy(15,15,#5'^PE'#5' เข้า-ออก'#5'ตัวขยาย'#5#2' ป้อนข้อความได้ตามต้องการ');
11:  setfillstyle(9,13);
12:  bar(0,0,getmaxx,10);
13:  Bar(0,270,getmaxx,280);
14:  Bar(0,0,12,280);
15:  Bar(getmaxx-12,0,getmaxx,280);
16:  readchar;
17:  closegraph;
18: end.
```

รูปที่ 7
ภาพหน้าจอแสดงตัวอย่างข้อความ (รหัส สมอ.) บนจอภาพ จากยูนิต thaitis.tpu



โปรแกรมตัวอย่างการใช้งาน

เพื่อให้เห็นรูปแบบที่แสดงผลบนจอภาพ จึงขอเขียนไฟล์ตัวอย่างชื่อ testtis.pas เพื่อแสดงผลบนจอภาพ โปรแกรมจะเป็นดังรูปที่ 6 และได้ภาพบนจอภาพดังภาพที่ 7 ซึ่งพิมพ์ด้วยเครื่องพิมพ์ 24 หัวเข็ม

หวังว่าต่อจากนี้ไปท่านที่มีความจำเป็นต้องใช้ภาษาไทยรหัส สมอ. ก็คงไม่ต้องกังวลอีกต่อไปแล้วว่าไม่มีภาษาไทยกราฟิกรหัสนี้ใช้ เพราะเพียงแต่ท่านสร้างยูนิต thaitis.tpu ตามบทความนี้ก็นำไปใช้งานได้แล้ว โดยท่านเพียงสร้างบางชุดคำสั่งเพิ่มเติมเช่น ชุดคำสั่งสำหรับป้อนข้อความ เป็นต้น พบกันใหม่...ในฉบับต่อไปด้วยเรื่องที่น่าสนใจเช่นเคย